RESERVATIONS AND TIMETABLING

Service scheduling characteristics

- Operations → **activities**
  - e.g, meetings to be attended by certain people, game to be played by two teams.
- Data:
  - Processing time → duration
  - Release time → earliest possible start time
  - Due date → latest possible end time
  - Weight → priority

Service scheduling characteristics

- **Resources:**
  - Classroom, hotel, rental car, stadium, operating room, plane, ship, airport gate, dock, railroad track, person (nurse/pilot)
- Synchronization of resources may be important
  - Need a plane and a pilot
  - Classroom, AV equipment, professor, students
- Each resource may have its own characteristics
  - Classroom: capacity, equipment, cost, accessibility
  - Truck: capacity, refrigeration, speed
  - Person: specialist (surgeon, nurse) with skills (languages)

Service scheduling characteristics

- **Activities** may have
  - Time windows
  - Capacity requirements/constraints
- **Resources** may have
  - Setup/transition time – runways at airports
  - Operator/tooling requirements
  - Workforce scheduling constraints
    - Shift patterns, break requirements
    - Union and safety rules

Differences from manufacturing

- Impossible to "store" goods
  - If a hotel room is not filled it is not possible to "get back" the lost time
- Resource availability often varies
  - May even be part of the objective function
- Saying "no" to a customer is common
  - "No available seats on that flight" (even if there are some!)
  - Try to book a restaurant for 8 PM

Reservation systems and timetabling

- Hotel rooms, car rentals, airline tickets (and classroom scheduling)
- Utilization of a resource for a given period of time
  - With slack: $p_j \leq d_j - r_j$
  - Without slack $p_j = d_j - r_j$
- May not be able to schedule all requests

Objectives

- Maximize profit
- Maximize resource usage
- Minimize number of rejected requests
- Minimize losses of rejected requests

Reservation systems and timetabling

1. Reservation Systems without Slack (timing of job fixed)
   - m parallel machines (resources), n jobs (activities) fixed in time; interval scheduling
2. Reservation Systems with Slack
   - m parallel machines, n jobs (more flexible in time)
3. Timetabling with Workforce Constraints
   - W identical resources in parallel, n jobs using one resource
4. Timetabling with Operator or Tooling Constraints
   - Can require more than one resource that are ***not*** identical.

Reservation Systems without Slack

- Reservation system with n activities, m resources
- Release date r_j, due date d_j,
- Activities have weight w_j or w_{ij}
- All activities and resources are independent
- No slack: $p_j = d_j - r_j$. If accepted, job starts at time r_j.

Weight is often equivalent to profit

- **Can we accept all activities? Do we need all resources?**
  - **Objective 1:** Maximize number of activities processed **OR**
  - **Objective 2:** Min. number of resources needed for all activities

Example: car rental

- Four types of cars: subcompact, midsize, full size, and sport utility
- Fixed number of each
- Resource = type of car
- Activity = customer requesting a car
- May have resource subsets
  - Rent a subcompact or midsize car
  - Some substitutability of resources

Example: car rental

- Reservation is for p_j days.
- Profit for car type i is π_i. (π_{ij} if it depends on costumer)
- Weight w_j
- Time is divided into H (integer) slots.
- x_{ij}: binary variable that is 1 if activity j is assigned to resource i
- J_t: set of activities that need a resource in slot t $[t-1, t]$

Feasibility problem

- Can we process every activity?
- Can we assign activities to resources such that activity j is assigned to a set M_j of resources?
- ➢ Relatively easy to solve

Problem formulation

- Integer Programming problem

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}x_{ij}$$

Maximize total profit

$$\sum_{i=1}^{m} x_{ij} \le 1, \quad j = 1,\ldots,n$$

Every activity is assigned to at most one resource

$$\sum_{j\in J_t} x_{ij} \le 1, \quad i = 1,\ldots,n, \;\; t = 1,\ldots,H$$

Each resource has only one activity per time slot

IP formulation

- In general the problem is NP-hard (when both p_j and w_{ij} are free)
- Two special cases with exact algorithms
  - each of these algorithms sorts the activities to increasing r_j
- **Case 1:** processing times $p_j = 1$
  - decompose into separate time units
  - assign at each time unit most valuable activities first
  - **it is an independent problem for each time slot!**

2: Identical weights and machines

- Assume $w_j = 1$ and $M_j = \{1,\ldots,m\}$ (all resources identical)
- **Objective**: maximize number of activities assigned
- No time decomposition, but still an exact, simple algorithm cab be applied:
  - Order activities in increasing order of release dates:

$$r_1 \le r_2 \le \ldots \le r_n$$

  - Let J be the set of already scheduled activities

Algorithm: max. activities assigned

Step 1:

- Set $J = \varnothing$ and $j = 1$

Step 2

- If a resource is available at time r_j, assign it to activity j, add activity j to J, and go to Step 4.
- Otherwise go to Step 3.

Step 3

- Let j^* be such that (maximum completion time):

$$C_{j^*} = \max_{k\in J}(C_k) = \max_{k\in J}(r_k + p_k)$$

Algorithm: max. activities assigned

- If $C_j = r_j + p_j > C_{j^*}$, do not include activity j in J and go to Step 4.
- Else, delete activity j^* from J, assign activity j to the resource freed and include activity j in J

Step 4

- If $j = n$ STOP,
- Else, set $j = j + 1$ and return to Step 2

Unlimited # of resources

- No slack, arbitrary processing times, equal weights, identical resources
- Infinitely many resources in parallel
- **Minimize the number of resources used**

- Easily solved
- Order jobs as before

Algorithm

- ❑ Assign activity 1 to resource 1
- ❑ Suppose first $j – 1$ activities have been processed
- ❑ Try to assign j^{th} activity to a resource in use
- ❑ If not possible assign to a new resource

➢ *Special case of the node coloring problem in graph theory.*

Node coloring problem

- ❑ Consider a graph with n nodes
- ❑ If an arc (j, k) connects nodes j and k they cannot be colored with the same color
- ❑ How many colors do we need to color the graph?

Number of colors needed = Number of resources needed

Reservation systems with slack

- ❑ Now allow slack $p_j \leq d_j - r_j$ (time slots)
- ➢ Trivial case
  - all processing times $p_j = 1$, identical weights, identical machines
  - Schedule constructed progressively in time
- ➢ Non-identical processing times, weights w_j
  - NP-hard →
  - No efficient algorithm →
  - Heuristic needed!

Composite dispatching rule

- ❑ Preprocessing: determine flexibility of activities and resources
- ❑ Dispatch least flexible activity first on the least flexible resource, etc.

- ν_{it} is the number of activities that may be assigned to resource i during interval $[t–1, t]$.
- $|M_j|$ is the number of resources in set M_j (suitable for activity j).

Priority indices

- ❑ Priority index for **activities**

$$I_j = f(w_j / p_j, |M_j|)$$

  - The higher w_j/p_j and the smaller $|M_j|$, the lower the index.
  - Activities can be ordered in decreasing order: $I_1 \leq I_2 \leq \ldots \leq I_n$
- ❑ Priority indexes for **resources** (examples for $[t, t+p_j]$)

$$g(\nu_{i,t+1}, \nu_{i,t+2}, \ldots, \nu_{i,t+p_j}) = \sum_{l=1}^{p_j} \nu_{i,t+l} \Big/ p_j \quad \text{or}$$

$$g(\nu_{i,t+1}, \nu_{i,t+2}, \ldots, \nu_{i,t+p_j}) = \max(\nu_{i,t+1}, \nu_{i,t+2}, \ldots, \nu_{i,t+p_j})$$

Algorithm

Step 0
- ❑ Calculate both priority indices
- ❑ Order activities according to activity priority index I_j

Step 1
- ❑ Set $j = 1$.

Step 2
- ❑ For activity j select the resource and time slot with lowest resource index $g(\nu_{i,t+1}, \nu_{i,t+2}, \ldots, \nu_{i,t+p_j})$
- ❑ Discard activity j if it cannot be assigned to any resource

Step 3

If $j = n$ STOP; otherwise set $j = j+1$ and return to Step 2

Example 9.3.2

- Consider seven activities and three resources:

Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p_j	3	10	9	4	6	5	3
w_j	2	3	3	2	1	2	3
r_j	5	0	2	3	2	4	5
d_j	12	10	20	15	18	19	14
M_j	{1,3}	{1,2}	{1,2,3}	{2,3}	{1}	{1}	{1,2}

Example 9.3.2 (cont.)

- Consider:

$$I_j = f\left(w_j / p_j, |M_j|\right) = \frac{|M_j|}{w_j / p_j}$$

- The indices for the activities are the following:

Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**
I_j	3	6.67	9	4	6	2.5	2

Example 9.3.2 (cont.)

- Factors v_{it} are:

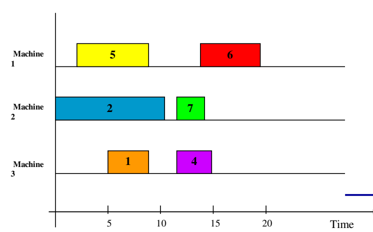
slot t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
v_{1t}	1	1	3	3	4	6	6	6	6	6	5	5	4	4	3	3	3	3	2	1
v_{2t}	1	1	2	3	3	4	4	4	4	4	3	3	3	3	2	1	1	1	1	1
v_{3t}	1	0	1	2	2	3	3	3	3	3	3	3	2	2	2	1	1	1	1	1

- Applying

$$g(v_{i,t+1}, v_{i,t+2}, \ldots, v_{i,t+p_j}) = \sum_{l=1}^{p_j} v_{i,t+l} \Big/ p_j$$

Example 9.3.2 (cont.)

- Yields the scheduling:



Activity	Resource	Period
7	2	11-14
6	1	14-19
1	3	5-8
4	3	11-15
5	1	2-8
2	2	0-10

Example 9.3.2: discussion

- Activity 3 does not fit into the schedule.
- Optimal schedule:
  - activity 7 starts with resource 1 at time 10
  - activity 3 starts with resource 2 at time 11
- Heuristic yields a suboptimal solution.
- Other functions for the priority indices for activities yield different schedules (see Pinedo's book).

Timetabling with Workforce Constraints

- Infinite identical resources in parallel
- n activities and all have to be done
  - Processing time of activity j is p_j
  - No preemption
- Total number of identical operators is W
- Each activity requires W_j operators
- Clearly, if $W_j + W_k > W$ then activity j and activity k cannot be processed at the same time.

Timetabling with Workforce Constraints

- ➢ Find a schedule that minimizes makespan
- ❑ Special case of Project Scheduling with Workforce Constraints with one resource, $p_j \geq 1$, no precedence

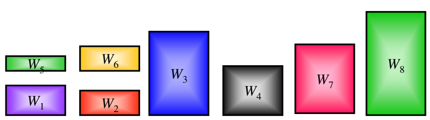
❖ Applications

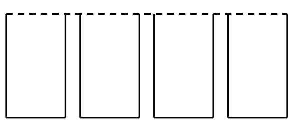
- scheduling a construction project (W is the crew size)
- exam scheduling (W is the number of seats)

Example 9.4.2: Exam scheduling

- ❑ All exams have the same duration
- ❑ One exam room with capacity W
- ❑ Course j has W_j students
- ❑ All students in course j must take the exam at the same time
- ❑ Find a timetable for all n exams in the minimum amount of time

Special case: bin packing

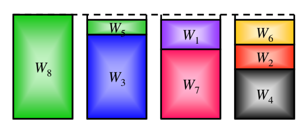




Pack the objects into the bins to minimize the number of bins used

Each bin has capacity W

Special case: bin packing



Pack the objects into the bins to minimize the number of bins used

Each bin has capacity W

Bin packing problem

- ❑ Assume equal processing times $p_j = 1$
- ❑ Unlimited number of machines
- ❑ Minimize makespan
- ❑ *Equivalent to bin packing problem*
  - each bin has capacity W and is one time slot
  - activity is an item of size W_j
  - items packed in one bin are activities done in time slot
  - pack into the minimum number of bins

Solving the bin packing problem

- ❑ Known to be NP-hard
- ❑ Many heuristics developed
- ❑ **First Fit (FF)** heuristic
  - Order activities (items) in an arbitrary order
  - Always put an item in the first bin it fits into
  - Known that

$$C_{\max}(\text{FF}) \leq \frac{17}{10} C_{\max}(\text{OPT}) + 2$$

Example 9.4.3

- Assume 18 activities and $W = 2100$.

activities	1,…,6	7,…,12	13,…,18
W_j	301	701	1051

- FF heuristic:
  - Assign the first 6 activities to one interval (301×6=1806)
  - Then assign two activities of 7,…,12 at a time to the next 3 intervals (701×2=1402)
  - Finally, assign one activity of 13,…,18 to each interval.

Example 9.4.3

- Makespan of FF: $C_{\max} = 10$.
- *Poor performance when jobs are assigned in arbitrary order*!

- **Optimal solution**: assigns to each slot of time three activities: one of 301, one of 701 and one of 1051 ($C_{\max} = 6$).

First Fit Decreasing (FFD)

- An improvement of FF.
- Order activities in decreasing order of W_j.

- Known that $C_{\max}(\text{FFD}) \leq \frac{11}{9} C_{\max}(\text{OPT}) + 4$

- *Find solution of Example 9.4.3 using FFD.*

- **FF and FFD can be extended to activities with different release dates.**

Example 9.4.4

- 30 activities and $W = 1000$.

activities	1,…,6	7,…,12	13,…,18	19,…,30
W_j	501	252	251	248

- Optimal schedule:
  - assign to each of the first 6 slots 3 activities: one of 501, one of 251 and one of 248.
  - To the remaining 3 slots it assigns 4 activities: two of 252 and two of 248 → $C_{\max} = 9$.

Example 9.4.4

- FFD rule:
  - assign to each of the first 6 slots 2 activities: one of 501 and one of 252.
  - To the next 2 slots it assigns three activities of 252.
  - To the remaining 3 slots it assigns four activities of 248 → $C_{\max} = 11$.

Timetabling with Operator Constraints

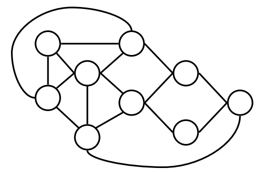
- Up to now, the W operators are equivalent
  - Any set of W_j operators could do activity j
- This is not true in many applications
  - e.g., medical or language specialties
- **Operator or Tool Constraints**
  - an operator need specific skills to do an activity
  - a specific tool is needed
- *If two activities require the same operator or tool, they cannot be done at the same time.*

Timetabling with Operator Constraints

- All n activities must be processed
- Minimizing makespan
- Timetabling is a special case of the *project scheduling with workforce constraints* problem:
  - no precedence constraints
  - One operator per skill (or one tool) $W_i = 1$.
- NP-hard
- Assume all activities durations = 1
- Equivalent to the **node (graph) coloring problem**

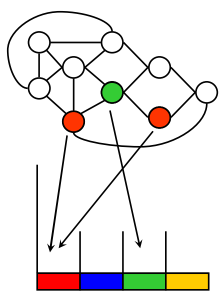
Node Coloring Problem

- Consider a graph with n nodes that are to be colored
- An arc (j,k) cannot receive same color.
- Color each node such that
  - No two connected nodes have the same color
  - Use the minimum number of colors



Node Coloring Problem

- Nodes are activities
- Arcs mean the activities require the same operator
- Colors are time slots
  - Minimizing the number of colors is minimizing makespan



Node Coloring Problem

- Recall
  - Activity = node
  - Activities need same tool = arc between nodes
- **Feasibility problem**:
  - Can the graph be colored with m (or less) colors?
- **Optimality problem**:
  - What is the lowest number of colors needed?

Heuristics for Node Coloring

- Terminology
  - **degree of node** = number of arcs connected to node
  - **saturation level** = number of colored nodes connected to node (in a partially colored graph)

- Many heuristics exist for graph coloring
- Intuition
  - Color high degree nodes first
  - Color high saturation level nodes first

Algorithm for Node Coloring

- **Step 1:** Order nodes in decreasing order of their degree.
- **Step 2:** Color node of maximal degree with Color 1.
- **Step 3:** Choose an uncolored node with maximum saturation level, breaking ties according to degree.
- **Step 4:** Color selected node using the color with the lowest possible number.
- **Step 5:** If all nodes are colored, STOP. Otherwise go to Step 3.

Example 9.5.2

- Four university professors have to attend several meetings.
- Goal: schedule all meetings from 2 to 6 p.m.

meetings	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Gary	1	0	0	1	1	0	1
Hamilton	1	1	1	0	0	0	0
Izak	0	0	1	0	1	1	0
Reha	1	0	1	1	1	0	0

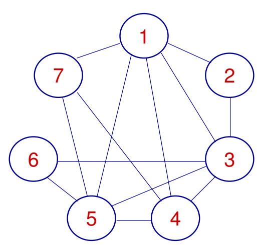
Example 9.5.2

- Problem can be transformed into a timetabling problem with operator constraints.
- Activities are nodes of the graph

Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**
operator 1	1	0	0	1	1	0	1
operator 2	1	1	1	0	0	0	0
operator 3	0	0	1	0	1	1	0
operator 4	1	0	1	1	1	0	0

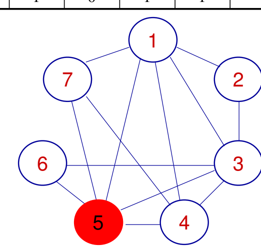
Corresponding graph

nodes	1	2	3	4	5	6	7
degree	5	2	5	4	5	2	3



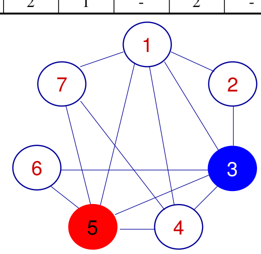
Color first node

nodes	1	2	3	4	5	6	7
degree sub	4	2	4	3	-	1	2
saturation	1	0	1	1	-	1	1



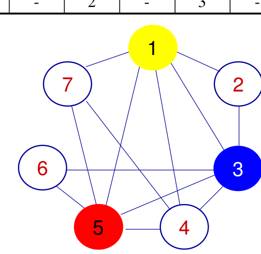
Color second node

nodes	1	2	3	4	5	6	7
degree	3	1	-	2	-	0	2
saturation	2	1	-	2	-	2	1



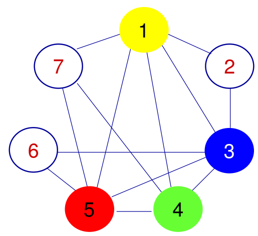
Color third node

nodes	1	2	3	4	5	6	7
degree	-	0	-	1	-	0	1
saturation	-	2	-	3	-	2	2



Color rest of the nodes

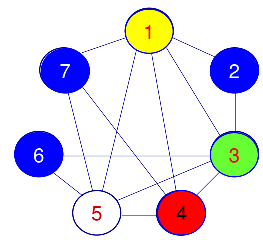
nodes	1	2	3	4	5	6	7
degree	-	0	-	-	-	0	0
saturation	-	2	-	-	-	2	3



Result 4 colors
7: blue
2: red
6: yellow

Reserse method is worse

nodes	1	2	3	4	5	6	7
degree	5	2	5	4	5	2	3



Another color needed for node 5
Result: 5 colors

Relation to reservation models

- Closely related to reservation problem with zero slack and arbitrary processing times
- Special case of timetabling problem
  - tools in common = overlapping time slots
  - tools in common ⇒ nodes connected
  - colors = resources
  - minimizing colors = minimizing resources
  - adjacent time slots vs. tools need not be adjacent

Timetabling and reservations

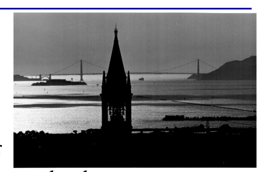
- Example of reservation system:

activities	*1*	*2*	*3*
p_j	2	3	1
r_j	0	2	3
d_j	2	5	4

Equivalent timetabling problem

	activities	1	2	3
Job 1 must be processed in time [0,2]	*Tool 1*	1	0	0
	Tool 2	1	0	0
Job 2 must be processed in time [2,5]	*Tool 3*	0	1	0
	Tool 4	0	1	1
	Tool 5	0	1	0

Example: classroom timetabling



- UC Berkeley
  - 30,000 students, 80 depts
  - 4000 classes, 250 rooms
  - 3 schedulers and 1 supervisor
- For each section of each course, the departments supply
  - Estimated enrollment
  - Requested meeting time
  - Special requirements (e.g., A/V equipment)

Development of an objective

Some obvious components:

- One class at a time in a given room, for a given prof, for a given number of students
- Usually minimize the number of students who cannot take the courses they want
- Room should be big enough
- Special equipment should be present

Development of an objective

Some non-obvious components:

- Profs want rooms close to their offices
- Students want consecutive classes to be close together
- Profs get one day with no classes
- (Departments want classes in rooms they "own")
- (Everyone wants no classes on Friday)

Development of an objective

- How do you balance the components?
  - Is a room within 100 m of a prof's office worth not being able to accommodate all students?
  - "You can have a Friday afternoon class with A/V equipment or a Friday morning class without."

Berkeley guidelines

- Standard "calendar"
  - 9-hour day, starting at 8 AM
  - 9 1-hour blocks overlap with 6 1.5-hour blocks
- "Prime time" blocks
  - One department can only request 60% of its classes during prime time.

Berkeley solution

- Large Integer Programming problem
  - 0.5M variables, 30K constraints
  - Very high penalty for not scheduling a class at all
  - Other objective components: distance, over-utilized facilities, empty seats
- **Solved heuristically!**
  - See Alg. 9.6.1, p 222

Room assignment heuristic

- J is the set of all classes
- t is a timeslot
- J_t is the set of all classes assigned to t
- M is the set of all classrooms
- M_j is the set of classrooms that can contain class j

Room assignment heuristic

Step 1 (*Select Time Slot*)

- Select, among time slots not yet considered, slot t with the smallest supply/demand ratio.

Step 2 (*Greedy Algorithm*)

- Rank all classes j in J_t in decreasing order of class size.
- Assign class j to the (vacant) room M_j with lowest cost

Step 3 (*Improvement Phase*)

- Rank all classes j in J_t in decreasing order of current cost.

Room assignment heuristic

- If class j is not assigned, find all interchanges with a class k in an occupied room, moving k to a vacant room. Make interchange with maximum cost reduction.
- If class j is assigned, find interchanges that reduce total cost. Apply interchange with maximum cost reduction.

Step 4 (*Stopping Criterion*)

- If Step 3 reduced the total cost, return to Step 3; otherwise update unscheduled list.
- If all time slots were selected, STOP; else go to Step 1.

Discussion

- This chapter considered four types of scheduling and timetabling problems.
- Real world problems have all the features discussed: release dates and due dates (with or without slack), workforce and tool constraints.
- In practice:
  - Dynamic rather than static reservation systems
  - Price considerations, which depend on current occupancy and forecast demand.